(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 014 266 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
28.06.2000 Bulletin 2000/26

(51) Int. Cl.[7]: **G06F 11/14**

(21) Application number: **99307546.4**

(22) Date of filing: **23.09.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **23.10.1998 US 178100**

(71) Applicant:
International Business Machines Corporation
Armonk, NY 10504 (US)

(72) Inventors:
- **Huang, Yun Wu,**
  c/o IBM United Kingdom Ltd.
  Winchester, Hampshire SO21 2JN (GB)
- **Yu, Philip Shi-Lung,**
  c/o IBM United Kingdom Ltd.
  Winchester, Hampshire SO21 2JN (GB)

(74) Representative:
Moss, Robert Douglas
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) Method, apparatus and program storage device for a client and adaptive synchronization and transformation server

(57) A sync proxy or sync server logic, process and apparatus employing sync logic and/or transformation codes distributed in the network to increase the scalability and improve the manageability of synchronization between a client such as: an Internet appliance; a handheld or palmtop device; a smart phone or intelligent pager; and their remote replica sources. An example of an adaptive sync server for synchronizing data between a client and a remote host replica, which stores a replica of data on the client includes sync acceptor logic for identifying a replica host and a sync logic which is application specific to a data type associated with the client and the remote host; wherein the sync logic can be located anywhere on a network remote to the sync server and the remote host, in response to a sync request; and sync handler logic, coupled to the sync acceptor logic, for retrieving remote sync logic from the network, connecting to the remote host, and synchronizing the data between the client and the remote host using the retrieved sync logic. The sync request could explicitly (or by default) identify the replica host and the sync logic or could include an address of a directory service storing information for identifying the replica host and the sync logic. The sync request could also or alternatively include information for identifying a (local or remote) device specific transformation code for converting data between the client and the remote host during synchronization; and means for retrieving the transformation code based on the information and converting the data between the client and the remote host based on the device specific transformation code.

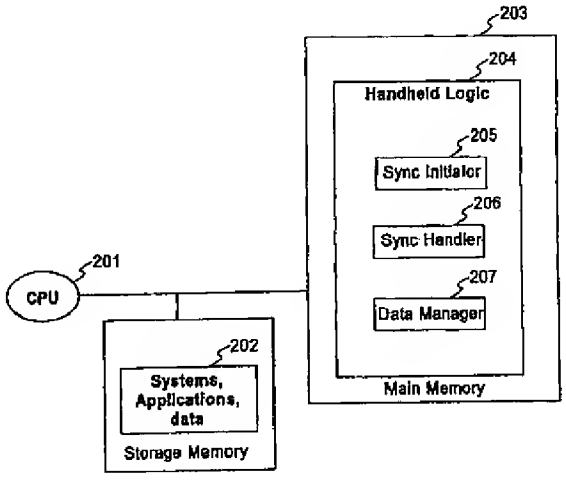

Fig.2

EP 1 014 266 A2

Description

[0001] The present invention relates generally to an update synchronization and/or transformation process between data in a client and replicas of the data in information management systems.

[0002] With the rapid advancement of semiconductor, storage and display technologies, handheld or mobile devices, which may only be occasionally connected to the network, have become increasingly popular. The ways that mobile devices such as 3Com's PALM PILOTTM are being used have also become very versatile. For example, a PALM PILOTTM user may have an e-mail application that is to be synchronized with a Lotus NOTESTM e-mail system in an office desktop. There may also be a stock quote application in the PALM PILOTTM which gets updated by synchronizing with a well-known stock quote site located in the Internet. There may also be a PALM PILOTTM expense application that is to be synchronized with home PC finance software. Therefore, it is quite possible that various handheld applications within one device need to synchronize with different applications in various information management systems (such as Lotus NOTESTM, Microsoft EXCHANGETM, file systems, relational databases, object-oriented databases) running on networked computers. It is also possible that one handheld application (such as e-mail) has different versions for different handheld devices (e.g., PALM PILOTTM, Sharp's ZAURUSTM, Psion PLC's PSIONTM and various other palmtop devices running Microsoft WINDOWS CETM).

[0003] Because handheld devices such as PALM PILOTsTM are typically only occasionally connected to the network with the connection lasting only a short time, it is critical that the synchronization processes between the applications in the handhelds and their counterparts in various networked computers be very efficient (so that the synchronization process can be successfully completed while the handhelds are connected to the network). With the handheld devices typically offering less CPU power and less memory for computation, the synchronization processes are usually not conducted inside the handheld devices.

[0004] Here, a computer hosting the replicas of data from handheld devices is called a replica host, or replica source. Because handheld devices generally have less computing power and memory than a replica host, the handheld devices typically conduct synchronization inside their replica hosts, which are desktops PCs. Typically, synchronization is first initiated by pushing a handheld device button (e.g., the HotSync button in a PALM PILOTTM). Immediately after the button is pushed, a synchronization manager software in the replica host (e.g., PALM PILOT'sTM HotSync Manager in the desktop personal computer (PC) takes over the synchronization process. In the case of a PALM PILOTTM, the HotSync Manager conducts synchronization for a PALM PILOTTM application by executing an application-specific program (called Conduit in the PALM PILOTTM) which synchronizes this application between the PALM PILOTTM and a desktop PC through application-specific logic. If multiple applications need to be synchronized, the HotSync Manager executes each of their corresponding Conduits in sequential order. Based on this approach, synchronization is always conducted between one PC and one handheld device. Simultaneous synchronization between multiple handheld devices and one PC is not available.

[0005] In general, conducting synchronization processes directly in the replica host is very limiting. The reason is that there may exist many applications (such as e-mail, calendar, address book) that each can be shared by many different handheld devices (e.g., PALM PILOTTM, ZAURUSTM, PSIONTM). Each application may require a different synchronization logic for each different device. To process synchronization for all applications and all devices properly, a replica host may need to install and manage many different versions of the synchronization processes in order to handle different handheld devices. Thus, conducting synchronization processes in the replica hosts creates a heavy burden in management of the different handheld devices and their synchronization processes. For many replica hosts, their most mission-critical task is not to synchronize with various handhelds, but to run as an information server such as a database server, a groupware server, or as a personal desktop computing device. Running synchronization on them therefore detracts from their mission-critical tasks.

[0006] A more efficient approach to management synchronization between multiple handheld devices and replica hosts is to deploy synchronization proxies (or synchronization (sync) servers, or synchronization proxy servers, or simply sync proxies) that maintain persistent network connections, to conduct synchronization on behalf of both the handhelds and the replica hosts (devices hosting the replicas also maintain a persistent network connection).

[0007] To synchronize, a handheld first establishes a network connection. Once connected, instead of making a direct connection to the targeted replica host for synchronization, it makes a synchronization request to a sync proxy. Upon receiving such a request, the sync proxy establishes a connection to the targeted replica host and starts the synchronization process by issuing input/output requests to both the handheld and its replica host. A straightforward design of the synchronization proxy is for the proxy to maintain all device-specific and application-specific synchronization programs that can handle all the handheld device types and all the information management system types hosted by the replica hosts. This means that each application must have an unique synchronization program written specifically for any combination between all the handheld device types and all the information management sys-

tems hosting the replicas of handheld data. For example, a synch proxy may maintain four different synchronization programs for an e-mail application in order to support both PALM PILOTTM and WINDOWS CETM based handheld devices and both Microsoft EXCHANGETM and Lotus NOTESTM e-mail systems for PCs. The first synchronization program is for synchronizing between PALM PILOTTM and EXCHANGETM, the second between PALM PILOTTM and Lotus NOTESTM, the third between WINDOWS CETM and EXCHANGETM, and the fourth between WINDOWS CETM and Lotus NOTESTM.

[0008] Another aspect of this straightforward design of the sync proxy is to allow the synchronization input/output interface between the synchronization proxy and a specific information management system hosting replicas of handheld data to be the same for different handheld devices. It is the sync proxy that handles the differences between handheld devices, not each replica host. The actual synchronization processes are conducted by the synchronization proxies. The role of both the handhelds and the replica hosts during synchronization is simply to respond to the synchronization input/output calls made by the synchronization proxy. These calls are typically in the form of adding, retrieving, deleting, and updating information. The computation burden and complexity of the synchronization logic for both handhelds and replica hosts therefore is much reduced.

[0009] One important feature of this straightforward sync proxy approach is that in order to properly synchronize between any applications for all device types and any information management systems for all replica hosts, a sync proxy must be able to invoke, during a synchronization task, a specific synchronization logic based on the target application, device, and the information management system. The current straightforward sync proxy approach accomplishes this by first restricting its support to a small number of applications, devices and information management systems to reduce the total number of synchronization logic variations required for proper synchronization; and by preconfiguring the synchronization logic as dynamically linkable (during synchronization runtime) codes stored in its disk memory. An important issue of such an approach is the difficulty in the management of all the various synchronization logic. If more applications, devices, and management information systems are supported by an organization, the number of sync logic variations increases dramatically. Each sync proxy therefore must be able to link the right sync logic. Furthermore, in an organization with multiple sync proxies deployed, any changes and updates to a sync logic creates a difficult software management task of propagating these changes and updates to the right files of all sync proxies. The improvement of the management of the synchronization logic is one focus of the present invention.

[0010] Another feature of the current straightforward sync proxy approach is that the various synchronization logic are pre-coded by the makers of the sync proxies, some of which also provide a development platform for users to augment or modify the synchronization logic. The Sync proxy makers do not generally create the applications, the devices, or the information management systems. Thus, they do not have the expert knowledge of this software and hardware and must limit the scope of their support. The scalability of current sync proxies in terms of supporting more handheld applications, more handheld devices, and more management information systems is very limited. The present invention addresses this limitation.

[0011] Furthermore, the straightforward sync proxy is a standalone system whose scalability in terms of processing simultaneous sync requests is limited by the computing power of the machine on which the sync proxy is installed. No collaboration between sync proxies in terms of load balancing is available in the straightforward sync proxy approach. This lack of collaboration between sync proxies may potentially create unbalanced loading among proxies which causes some sync requests to be delayed or not serviced. The present invention addresses this need.

[0012] In accordance with the aforementioned needs, the present invention is directed to an improved method, apparatus and program storage device for a synchronization proxy (or sync proxy or sync server) that serves as an intermediary and a computation platform in performing update synchronization between clients, examples of which include but are not limited to: an Internet appliance; a handheld or palmtop device; a smart phone or intelligent pager; and information management systems that host the replicas of data from these devices. The clients may be disconnected from the network some or most of the time while the information management systems remain connected almost all the time. The data in the client devices and/or their replicas in the information management systems may be updated independently of each other.

[0013] According to a first aspect, the present invention provides an adaptive sync server for synchronizing data between a client and a remote host replica, which stores a replica of data on the client, as claimed in claim 1.

[0014] Preferably, the sync acceptor logic further comprises means for receiving from the client the sync request including information for identifying the replica host and the remote sync logic; the sync logic comprises means for one or more of: resolving a conflict, specifying an overriding direction, and taking certain actions to fulfil a specific need of the application; the sync request includes an address of a directory service storing information for identifying the replica host and the sync logic; and the sync handler logic further comprises means for retrieving the information, in response to the sync request.

[0015] Preferably, the sync request includes information for identifying a device specific transformation code for converting data between the client and the remote host during synchronization; and the sync handler logic comprises means for retrieving the transformation code based on the information and converting the data between the client and the remote host based on the device specific transformation code; and wherein the device specific transformation code can be remotely located anywhere on the network, and wherein the sync request includes an address of a directory service storing information for identifying the device specific transformation code for transforming the data between the client and the remote host; and the sync handler logic comprises means for retrieving remote transformation code based on the information and converting the data between the client and the remote host based on the device specific transformation code, and wherein the transformation code comprises means for one or more of filtering; transforming; and changing the data so that it can be used and processed in different computing devices having one or more of different CPU powers, memory capacities, and physical constructs.

[0016] Preferably, the server further comprises a load manager, coupled to the acceptor logic, for monitoring a server computation load and a server computation capacity and sharing the computation load and the computation capacity with collaborating sync servers; and said load manager comprising logic for diverting the sync request to a less loaded server, when the server computation load exceeds a threshold. Further, the network includes one or more of an intranet and Internet wherein the transformation code further comprises means for filtering an image or transforming a resolution of the image included in a web page sent from the remote host to the handheld device so that the page can be shown on a screen on the handheld device within the memory constraints of the handheld device.

[0017] Preferably, the server further comprises cache manager means, coupled to the sync handler, for retrieving and storing the sync logic into a cache memory, wherein said cache manager means further comprises means for presetting and storing the sync logic into the cache memory.

[0018] Preferably, identifying information comprises a URL. Further, the network is the world wide web and the sync server is a proxy server and wherein the client is selected from a group consisting of handheld device; a smart phone, or an intelligent pager.

[0019] According to a second aspect, the invention provides an adaptive server for transforming data between a client and a remote host replica, which stores a replica of data on the client, as claimed in claim 2.

[0020] Preferably, the acceptor logic further comprises means for receiving from the client the request including information for identifying the replica host and the remote transformation code; wherein the request includes an address of a directory service storing information for identifying the replica host and the transformation code; and the handler logic further comprises means for retrieving the information, in response to the request; wherein the transformation code comprises means for one or more of filtering; transforming; and changing the data so it can be used and processed in different computing devices having one or more of different CPU powers, memory capacities, and physical constructs; wherein the transformation code further comprises means for filtering an image or transforming a resolution of the image included in a web page sent from the remote host to the handheld device so that the page can be shown on a screen on the handheld device within the memory constraints of the handheld device; wherein the request includes information for identifying a logic which is application specific to a data type associated with the client and the remote host, in response to a request; and the handler logic comprising means for retrieving the logic, connecting to the remote host based on the information, and synchronizing the data between the client and the remote host.

[0021] Further preferably to the latter, wherein the logic comprises means for one or more of: resolving a conflict; specifying an overriding direction; and taking certain actions to fulfil a specific need of the application, wherein the logic can be remotely located anywhere on the network, wherein the request includes an address of a directory service storing information for identifying the logic; and the handler logic comprises means for retrieving remote logic based on the information and synchronizing the data between the client and the remote host based on the logic.

[0022] Preferably, the server further comprises a load manager, coupled to the acceptor logic, for monitoring a server computation load and a server computation capacity and sharing the computation load and the computation capacity with collaborating servers; and said load manager comprising logic for diverting the request to a less loaded server, when the server computation load exceeds a threshold.

[0023] Preferably, the server further comprises cache manager means, coupled to the handler, for retrieving and storing the logic into a cache memory, wherein said cache manager means further comprises means for presetting and storing the logic into the cache memory.

[0024] Preferably, identifying information comprises a URL. Further, the network is the world wide web and the server is a proxy server and wherein the client is selected from a group consisting of handheld device; a smart phone, or an intelligent pager.

[0025] According to a third aspect, the invention provides a client device adapted for connection to a sync server wherein synchronization of data is performed between the client and a remote host replica storing a replica of the data on the client, as claimed in claim 3.

[0026] Preferably, the client is a disconnectable

handheld device selected from a group consisting of handheld computer; a smart phone, or an intelligent pager, comprising means for establishing a connection to the network, wherein the data management comprise one or more of reading a data item; writing a data item; updating a data item; and deleting a data item, further comprising means for communicating version and update history information between the client and the remote replica host, wherein the sync request further comprises information for identifying the replica host and the remote sync logic, wherein the sync request includes an address of a directory service storing information for identifying the replica host and the sync logic, wherein the sync request includes information for identifying a device specific transformation code for transforming data between the client and the remote host, wherein the device specific transformation code can be remotely located anywhere on the network, wherein the sync request includes an address of a directory service storing information for identifying one or more of sync logic and the device specific transformation code for transforming the data between the client and the remote host, wherein the client is a handheld device disconnectably coupled to one or more of an intranet and Internet wherein the data includes an image included in a web page sent from the remote host to the handheld device transformed so that the page can be shown on a screen of the handheld device within the memory constraints of the handheld device.

[0027] Preferably, identifying information comprises a URL.

[0028] Preferably, the request includes one or more network addresses of the sync server stored in the memory wherein one sync server address is pre-configured as a default sync server address.

[0029] Preferably, the request includes identification information about the client or a user selected from a group consisting of one or more of : a user ID; an encrypted password; shared secret information for authentication and authorization; a device type; or a system type associated with the client.

[0030] Preferably, the sync initiator is adapted for determining a list of applications needed to be synchronized, said applications selected from a group consisting of one or more of : an address book application; a memo pad application; a calendar application; an e-mail application.

[0031] According to a fourth aspect, the invention provides a client device adapted for connection to an adaptive server wherein transformation of data is performed between the client and a remote host replica storing a replica of the data on the client, as claimed in claim 6.

[0032] Preferably, the client is a disconnectable handheld device selected from a group consisting of handheld computer; a smart phone, or an intelligent pager, comprising means for establishing a connection to the network, wherein the request further comprises information for identifying the replica host and the transformation code, wherein the request includes an address of a directory service storing information for identifying the replica host and the transformation code, wherein the request includes information for identifying a logic which is application specific to a data type associated with the client and the remote host, wherein the logic can be located anywhere on a network remote to the server and the remote host, wherein the request includes an address of a directory service storing information for identifying one or more of the logic and the device specific transformation code. Preferably, the client is a handheld device disconnectably coupled to one or more of an intranet and Internet wherein the data includes an image included in a web page sent from the remote host to the handheld device transformed so that the page can be shown on a screen of the handheld device within the memory constraints of the handheld device.

[0033] Preferably, identifying information comprises a URL.

[0034] Preferably, the request includes one or more network addresses of the server stored in the memory wherein one server address is pre-configured as a default server address.

[0035] Preferably, the request includes identification information about the client or a user selected from a group consisting of one or more of : a user ID; an encrypted password; shared secret information for authentication and authorization; a device type; or a system type associated with the handheld.

[0036] Further, the initiator is adapted for determining a list of applications needed to be synchronized, said applications selected from a group consisting of one or more of : an address book application; a memo pad application; a calendar application; an e-mail application.

[0037] According to a fifth aspect, the invention provides, in an adaptive server, a method for adaptively synchronizing data between a client and a remote host replica storing a replica of data on the client, as claimed in claim 7. The preferable features discussed above with respect to the corresponding apparatus claim 1 are also relevant for this fifth aspect.

[0038] According to a sixth aspect, the invention provides, in an adaptive server, a method of transforming data between a client and a remote host replica storing a replica of data on the client, as claimed in claim 8. The preferable features discussed above with respect to the corresponding apparatus claim 2 are also relevant for this sixth aspect.

[0039] According to seventh and eighth aspects, the invention provides program storage devices readable by a machine, tangibly embodying a program of instruction executable by servers, as claimed in claims 9 and 10, to carry out the methods corresponding to the fifth and sixth aspects mentioned above.

[0040] According to a preferred embodiment of the

present invention, a method is provided for a client to request synchronization services from a sync proxy. For example, a handheld device issues a synchronization request (or sync request) to this proxy. A sync request may involve one or more applications to be synchronized. After the sync request is accepted, the sync proxy processes synchronization for all applications requested to be synchronized for this handheld device in a sequential fashion. For each application to be synchronized, the handheld may provide a sync identifier which may include the name of the application to be synchronized, the ID of the replica host for this application, the ID of the program with application specific synchronization logic for this application (sync logic), and the ID of the program with a device specific data transformation method for this handheld device (transformation code) . The IDs preferably includes a unique identifier (or a unique name) and a network address from which this information can be retrieved.

[0041] Alternatively, during the sync processing of an application, instead of the handheld device sending the information directly to the sync proxy, the handheld device may send a sync identifier which may only include some identification information of the user, the handheld device, and the application to be synchronized, as well as the address of a directory service in which the name of the application and the aforementioned three pieces of information (i.e., the ID of the replica host for this application, the ID of the sync logic for this application, and the ID of the transformation code for this handheld device) is stored.

[0042] An example of an adaptive sync server for synchronizing data between a client and a remote host replica, which stores a replica of data on the client, having features of a preferred embodiment of the present invention includes: sync acceptor logic for identifying a replica host and a sync logic which is application specific to a data type associated with the client and the remote host; wherein the sync logic can be located anywhere on a network remote to the sync server and the remote host, in response to a sync request; and sync handler logic, coupled to the sync acceptor logic, for retrieving remote sync logic from the network, connecting to the remote and synchronizing the data between the client and the remote host using retrieved sync logic. The sync request can also include information for identifying a device specific transformation code for converting data between the client and the remote host during synchronization; wherein the sync handler logic includes means for retrieving the transformation code based on the information and converting the data between the client and the remote host based on the device specific transformation code. Here, the device specific transformation code can be locally or remotely located anywhere on the network.

[0043] Another example of an adaptive sync server for transforming data between a client, and a remote host replica storing a replica of data on the client, having features of a preferred embodiment of the present invention includes: sync acceptor logic for identifying the remote replica host and a device specific transformation code for transforming the data on the remote host to that of a device type associated with the client, wherein the transformation code can be located anywhere on a network remote to the sync server and the remote host, in response to a request; and sync handler logic, coupled to the sync acceptor logic, for retrieving remote transformation code and transforming the data between the client and the remote host based on the device specific transformation code. The request can include a sync request for identifying a sync logic, which is application specific to a data type associated with the client and the remote host, in response to a sync request; and means for retrieving the sync logic, connecting to the remote host and synchronizing the data between the client and the remote host during the transformation.

[0044] According to a preferred embodiment of the present invention, a method is provided for the sync proxy to receive, accept, and process sync requests made from the handheld devices. In the preferred embodiment, a sync proxy - upon receiving a sync request from a handheld device - conducts synchronization and/or transformation for one or more handheld applications. For each handheld application, the proxy may receive the aforementioned name of the application, the replica host ID, sync logic ID, and transformation code ID either directly from the requesting handheld device or indirectly from a directory service specified by the requesting handheld device. To synchronize an application, the sync proxy may retrieve the sync logic associated with this application from the network address specified in the sync logic ID for this application sent by the requesting handheld device. The sync proxy then establishes a network connection with the replica host associated with this application based on its replica host ID received from the requesting handheld device.

[0045] According to a preferred embodiment of the present invention, to process synchronization for this application, the sync proxy executes the retrieved sync logic for this application. During the execution of the sync logic for this application, if data transformation is required, the sync proxy makes a connection to the network address of the transformation code associated with this application. The network address of the transformation code is part of the transformation code ID sent by the requesting handheld device. Then the sync proxy processes data transformation if necessary by executing this transformation code during the processing of the sync logic while synchronizing this application. While the present invention does not specify which programming language a sync logic is written with, the feature of the present invention where the sync proxy retrieves a sync logic from a remote host and executes this logic can be thought of in terms of existing Java enabled web processing where a Java program such as

an applet can be downloaded by a web browser from a remote host and executed by the web browser locally for web processing.

[0046] In one embodiment of the present invention, the synchronization procedure for an application is divided into sync logic and transformation code. The sync logic is the application specific procedure of the synchronization tasks whereas the transformation code is the device dependent conversion process between two types of devices. The separation of these two means that the sync logic for an application can be provided and maintained by the maker of this application whereas the transformation code can be provided and maintained by the maker of the handheld device to which this transformation code applies. The sync proxies no longer have to store and maintain these synchronization or transformation procedures. All they need to do is retrieve the proper sync logic and/or transformation codes during synchronization. This approach greatly improves the management process of the application dependent and device dependent synchronization and/or transformation procedures by the sync proxies.

[0047] Such an approach also relieves the makers of the sync proxies from having to develop the synchronization procedures that are specific to the applications and the devices. As a result, the sync proxies can be more scalable in terms of supporting more applications, devices, and management information systems, as long as the application and device dependent sync procedures are properly developed and maintained and made retrievable by their respective makers.

[0048] As a preferable feature of a preferred embodiment of the present invention, a sync proxy may deploy a cache to temporarily store the sync logic or the transformation code for an application in anticipation that these executable data may be used by this sync proxy in the near future. An example of the cache feature of the present invention is described below. First, the sync proxy sets aside a block of memory (either main memory or disk space) as the cache. The sync proxy may adopt an indexing method to search and retrieve a cached sync logic or transformation code (e.g., based on their unique names). To retrieve a cached information (the sync logic or the transformation code for an application), the sync logic, before going out to the network to retrieve it, first searches its cache. If the information is not found in the cache, the sync proxy then goes to the network address associated with the ID of this information to retrieve it and updates the proxy's cache with it. If the information is found in the cache, the sync proxy retrieves this information from its cache instead of from the network. Then, the sync proxy executing this executable data (the sync logic or the transformation code for an application) by loading this data from the cache for execution.

[0049] As another preferable feature of a preferred embodiment of the present invention, a sync proxy can continuously monitor its computation load (e.g., in terms of the number of sync requests it is concurrently processing) against its computation capacity which is fixed based on its CPU power, the size of its RAM and storage memory, and its network bandwidth capacity. All synchronization proxies within the same network can participate in a real-time sharing with one another of their current computation load and their pre-configured computation capacity. Upon receiving a sync request, a sync proxy checks to see if the addition of this synchronization task to the current computation load exceeds the computation capacity of this proxy. If so, this proxy queries other proxies in the same network about their computation load and capacity information. This proxy, after getting the information, can then divert this incoming sync request to another proxy (in the same network) whose computation load is less. If all proxies on the same network have a full computation load, then the proxy that receives this sync request can return to the requesting handheld device a message indicating that the capacity for all sync proxies are full. It also terminates the connection with the requesting handheld device. Synchronization processes are not performed in this case. With this feature, the present invention also provides a load balancing function not available in the straightforward sync proxy approach.

[0050] These, and further, objects, advantages, and features of the invention will be more apparent from the following detailed description of a preferred embodiment and the appended drawings wherein:

Figure 1 depicts an example of an architecture of the sync proxy approach;

Figure 2 depicts an example of an architecture of a client implemented as a handheld device having features of the present invention;

Figure 3 depicts an example of an architecture of a sync proxy of the present invention;

Figure 4 depicts an example of an architecture of a replica host feature of the present invention;

Figure 5 depicts an example of a handheld sync acceptor feature of the present invention;

Figure 6 depicts an example of a proxy sync acceptor feature of the present invention;

Figure 7 depicts an example of a handheld sync handler feature of the present invention;

Figure 8 depicts an example of a proxy sync handler feature of the present invention;

Figure 9 depicts an example of a proxy cache manager feature of the present invention;

Figure 10 depicts an example of a proxy sync handler processing data transformation;

Figure 12 depicts an example of a replica manager in host replica or a data manager in a handheld device; and

Figure 13 depicts an example of a proxy load manager feature of the present invention.

[0051] Figure 1 depicts an example of an overall architecture of a network deploying the synchronization server approach having features of the present invention. A client (101, 102), examples of which include but are not limited to: handheld or palmtop devices (also called a hand-held terminal, palmtop computer, Internet appliance). A handheld device generally refers to any computer-based device small enough to be held in one hand while being operated by the other. Handheld devices may have smaller displays because the nature of the work for which they are designed does not require a great amount of information to be shown at one time. Handheld devices commonly contain communications equipment which allow them to communicate and/or synchronize with a central computer. Examples of handheld devices include but are not limited to 3Com's PALM PILOTTM, Sharp's ZAURUSTM, Psion PLC's PSIONTM and various other "palm" type devices running Microsoft WINDOWS CETM, a smart phone, or an intelligent pager, etc. Thus, although the preferred embodiment refers to handheld devices, one skilled in the art will appreciate that the present invention can be beneficial for any client device which synchronizes data over a network.

[0052] The clients can be intermittently connected to servers (105-107) (also called synchronization proxy, sync proxy, sync server, or sync proxy server). Examples of the servers (105, 106, 107) include but are not limited to: a PC; a workstation (such as an IBM RS6000TM); or a mainframe computer. A replica host (109, 110, 111) can be any computer running an information management system which maintains replicas of data from the handheld devices. A replica host can be a PC, a workstation, or a mainframe, etc. The sync proxies are connected with the replica hosts through the network (108). The handheld devices can be unconnected to the network some or most of the time, but have to remain connected (103, 104) while they are performing synchronization tasks. Those skilled in the art will appreciate that a handheld device can obtain a connection to a network (such as the Internet or a local area network) by dialling up to a network remote access server through a modem, or by having a direct serial-port connection (e.g., using the PALM PILOTTM cradle) to a computer (such as a desktop PC) that is connected to the network. As is conventional, a directory server (112) provides services including the pre-configuration and storage of information for users and provides a search engine to dynamically retrieve the information through a network such as the Internet upon request. Examples of these services include Novell's Novell Directory Services (NDS) and Microsoft's Active Directory both of which provide directory information which can be accessed through LDAP (Lightweight Directory Access Protocol - an Internet protocol for accessing directory information).

[0053] According to the present invention, sync logic (120) and/or transformation code (130) are stored remotely. An example of an adaptive sync server for synchronizing data between a client (101-102) and a remote host replica (109-111), which stores a replica of data on the client, includes: sync proxy logic (Figure 3) for identifying the replica host (109-111) and the sync logic (120), which is application specific to a data type associated with the client and the remote host. The sync logic (120) can be located anywhere on a network remote to the sync server (105-107) and the remote host (109-111), in response to a sync request. The sync proxy logic is adapted for retrieving the remote sync logic from the network (108); connecting to the remote host based on the request; and synchronizing the data between the client and the remote host using retrieved sync logic. The sync request can also include information for identifying a device specific transformation code (130) for converting data between the client and the remote host during synchronization, including means for retrieving the transformation code based on the information and converting the data between the client and the remote host based on the device specific transformation code. Here, the device specific transformation code can be locally or remotely located anywhere on the network

[0054] Alternatively, the server is adapted for transforming data between the client (101-102), and a remote host replica (109-111) storing a replica of data on the client. Here, the request is used to identify the applicable remote replica host and a transformation code (130) for transforming the data on the remote host to that of a device type associated with the client. The transformation code can be located anywhere on a network remote to the sync server and the remote host. The server retrieves the remote transformation code (130) based on the request and transforms the data between the client and the remote host based on the device specific transformation code. The request can include a sync request with information for identifying the sync logic (120), which is application specific to a data type associated with the client and the remote host, in response to the sync request. The server retrieves the sync logic; connects to the remote host based on the information; and synchronizes the data between the client and the remote host during the transformation.

[0055] Figure 2 depicts an example of an overall architecture of a client implemented as a handheld device capable of performing synchronization in accord-

ance with the present invention. As depicted, the handheld device includes a CPU (201), a main memory (203) such as volatile RAM, and storage memory using (202) for example nonvolatile RAM, ROM, or disk for storing systems (such as the operating system), applications (such as the e-mail and calendar software), and the data (such as the content of an address book or memo pad). Most handheld devices store all information in RAM and ROM without any disks. The main memory (203) stores the handheld device logic (204) of the present invention that is preferably embodied as computer executable code which may be loaded from the storage memory (202) into the main memory (203). Here, the handheld logic (204) of the present invention (204) includes a sync initiator (205) (with details depicted in Figure 5), a sync handler (206) (with details depicted in Figure 7), and a data manager (207) (with details depicted in Figure 11). The handheld logic (204) of the present invention is initially stored in the storage memory (202). When the synchronization task is started, the CPU (201) loads the sync logic (either in full or in part in an on-demand fashion) into the main memory (202) and starts executing this logic for synchronization.

[0056] Figure 3 depicts an example of an architecture of a computing device configured as a synchronization proxy of the present invention. Examples include but are not limited to: a PC; a workstation; a server; or a mainframe. The sync proxy can include a CPU (301), storage devices (302) such as disks, and a main memory (303) such as RAM. Here, the main memory (303) stores the sync proxy logic (304), which is preferably embodied as computer executable code which may be loaded from the disks (302) into the main memory (303). In this example, the sync proxy logic (304) includes a sync acceptor (305) logic (with details depicted in Figure 6), a sync handler (306) logic (with details depicted in Figure 8), a cache manager (307) (with details depicted in Figure 9), and a load manager (308) (with details depicted in Figure 13).

[0057] Figure 4 depicts an example of an architecture of a computing device configured as a replica host of the present invention. Examples of a replica host include but are not limited to: a PC; a workstation; a server; or a mainframe. As depicted, the replica host can include a CPU (401). storage devices (402) such as disks, and main memory (403) such as RAM. The main memory (403) stores the replica host logic (404), which is preferably embodied as computer executable code which may be loaded from disks (402) into main memory (403). The replica host logic (404) includes a replica acceptor (405) (with details depicted in Figure 12) and a replica manager (406) (with details depicted in Figure 11).

[0058] As is conventional, the handheld device needs to first establish a network connection before the synchronization tasks can be performed. There are many available technologies in connecting a handheld device to the network such as the Internet or a local area network. For example, a PALM PILOTTM has a TCP/IP software that comes with the device. When a PALM PILOTTM is attached to a modem, it can run this TCP/IP software to dial up to an ISP (Internet Service Provider) through a phone line to get a connection to the Internet. Alternatively, the device can be connected through a serial port to a desktop PC that is connected to the network. There are programs available (such as Microsoft's Remote Access Service, or RAS) to enable a PC as a network access server and provide a connection to the network to any device that connects to this PC's serial port. While not part of the present invention, the network connection process of the client device is a prerequisite of the synchronization methods embodied in the present invention.

[0059] After the device establishes the connection to the network, this device can now perform the synchronization tasks by starting the sync initiator process depicted by the sync initiator (205) in Figure 2, and, in more detail in Figure 5. To start the synchronization process, the sync initiator first sends a sync request to a sync proxy (501). Those skilled in the art will appreciate that the network addresses of the sync proxies can be known beforehand by the handheld devices, and one sync proxy address can also be pre-configured as the default by the handheld devices. The information that the handheld device sends to the proxy to request synchronization can include a mutually understood code to indicate that it is a sync request, as well as some identification information about the user (such as the user ID and the encrypted password or shared secret for authentication and authorization) and the device and system type of the handheld. While the method for authentication and authorization is not a part of the present invention, those skilled in the art will appreciate that existing authentication and authorization technologies over the network can be incorporated into the collaborative procedure between the sync initiator of the handheld (501 in Figure 5) and the sync acceptor of the sync proxy (305 in Figure 3).

[0060] If the sync request is rejected, the sync initiator process terminates the sync task (504). Otherwise, the sync initiator determines the list of applications needed to be synchronized (502). The applications to be synchronized may include for example an address book application, a memo pad application, a calendar application, an e-mail application, or any other applications in the handheld device. Those skilled in the art can understand that determining a list of applications to be synchronized may involve a screen input from the user (such as checking the applications to be synchronized from a user interface), a retrieval of a default set of applications, or a software to dynamically selecting the applications to synchronize. Once the list of applications to be synchronized is determined, the sync initiator process starts a sync handler process (503) (with details depicted in Figure 7) for each application on the list. The first task of the sync proxy is to start the sync

acceptor (305 in Figure 3 and described in more detail with reference to Figure 6).

[0061] Figure 6 depicts an example of a proxy sync acceptor feature of the present invention. As depicted. The sync acceptor can be an infinite loop in which the sync proxy continuously receives sync requests from the handheld devices (601). Upon receiving a sync request, the sync acceptor may first check if its own computation load has reached its capacity (602). If so, the sync acceptor starts the load manager (603) to forward this request to another sync proxy on the same network (details depicted in Figure 13). If the sync proxy is not overloaded, the sync acceptor can perform a conventional authentication and authorization procedure based on the identification information inside the request messages received from the requesting handheld devices. If the authentication and authorization procedure fails (e.g., the handheld user failed to authenticate or is not authorized for synchronization by this sync proxy), the proxy sends a rejection message back to the handheld (605), terminates the connection (606) and goes back to wait for the next sync request (601). If the authentication and authorization procedure succeeds, the proxy then sends an acceptance message back to the requesting handheld device (607). Next, the proxy starts the proxy sync handler to synchronize for this synchronization task (608).

[0062] Those skilled in the art will appreciate that the sync acceptor process is preferably implemented with a multithread approach using current software technology. This means that upon receiving a request from a handheld (601), the sync acceptor starts a new thread to process the remaining steps (602 - 608) for this request. The three arrow-headed lines, from 603 to 601, from 606 to 601, and from 608 to 601, simply indicate a termination of this thread. Those skilled in the art will also appreciate that in an operating system that does not support multithreading, the creation of a new thread (601) can be thought of as creating of a new process. The termination of a thread therefore is equivalent to the termination of a process.

[0063] Returning now to the handheld devices, with reference also to Figure 5, the sync handler for each of the application to be synchronized is started in sequence (503) after the acceptance message is received from the proxy (605). Figure 7 depicts a more detailed example of the sync handler logic (for an application).

[0064] As depicted in Figure 7, the sync handler for a specific application first retrieves information associated with this application (701). They preferably include the ID of the replica host, the ID of the sync logic, and the ID of the transformation code for this application (701).

[0065] A replica host for an application is the computer device that stores the replica of this application. The device can be a PC, a workstation, server or a mainframe, and must be network connected. The ID of the replica host can include the network address of the replica host and the location of the replica of this application inside the replica host. Those skilled in the art will appreciate that, based on current Internet technology, a URL (Universal Resource Locator) is a reasonable way of implementing the replica host ID.

[0066] The sync logic for an application, preferably embodied as executable code, can be used to resolve conflicts, to specify overriding direction, or to take certain actions to fulfil the specific needs for this application. For example, the sync logic for an expense keeping application may specify that the handheld always overrides the replica host (such as a corporate database server) for certain records (such as hotel expenses) as the corporate records represent suggested prices whereas the handheld records specify the actual billings. The ID of the sync logic can include a unique name of the sync logic, the network address and the location inside this address where this sync logic is stored. Those skilled in the art will also appreciate that, using existing current Internet technology, a URL is again a reasonable way to implement the sync logic ID.

[0067] The transformation code for a specific device and application combination, preferably embodied as executable code, can be used to filter, transform, and change data so that certain information can be feasibly used and processed in computing devices having different CPU powers, memory capacities, and physical constructs. For example, a web page with images being sent from a PC replica host to a client may run a transformation code to have all images either removed (filter function) or reduced to a very coarse resolution (transform function) so that this page can be shown on the client's screen without taking up too much memory. The ID of the transform code for an application can include a unique name of the transformation code, the network address and the location inside this address where the transformation code is stored. Those skilled in the art will again appreciate that, using existing Internet technology, a URL is yet again a reasonable way to implement the transform code ID.

[0068] According to another feature of the present invention, the aforementioned three IDs for each application is pre-configured and stored in the handheld devices or in a directory service for the user of this handheld device. If these IDs for the application are stored in the handheld device, the sync handler sends the sync identifier, which preferably includes the name of the application to be synchronized together with the three IDs to the sync proxy (701) and then goes into a loop (702). If these IDs are stored by way of a directory service, the sync handler sends the sync identifier which includes the identification of the user, the device, the application, and the address of the directory server (with which the sync proxy will use to access the directory service) to the sync proxy (701).

[0069] There skilled in the art will appreciate that the need to send any of the three IDs can be eliminated

by the use of default settings. For example, if a handheld device configures with a sync proxy a default application to be synchronized, then this sync proxy need not retrieve the name of the application to be synchronized during the sync processing for this handheld device.

[0070] The lack of an application name in this case indicates the default application is to be synchronized.

[0071] Similarly, the user of a handheld device can also configure with a sync proxy a default replica host or device. In these cases, the sync proxy needs not retrieve this information either from the handheld device or from the directory. Instead, it can uses its pre-configured default setting for this handheld device to get this information.

[0072] While the ID of information such as the replica host, the sync logic, or the transformation code is described previously as including a network address and the exactly name and location with their respective address, an alternative implementation for an ID does not require the specification of its corresponding network address. Here, it is the sync proxy's responsibility to find out the exact network address of a particular ID based on the name included in this ID. An example is that a handheld device makes a sync request for an application where the ID for the transformation code does not include the network address of the proper transformation code for this device. The sync proxy, upon receiving the ID of the transformation code, determines the proper network location where the proper transformation code resides based on the type of the device that makes the sync request.

[0073] Referring again to Figure 7, within the loop (702), the sync handler first waits for an API (Application Program Interface) call from the sync proxy (702). If the call from the proxy indicates that the synchronization process for this application is completed, the sync handler terminates and goes back to the sync initiator to process the next application to be synchronized. If the API call is a data management function (such as opening a database, reading a record, etc.), the sync handler starts the data manager to process this API (703) (with more detailed discussions in Figure 11).

[0074] The sync acceptor (Figure 6) in the synchronization proxy - after authentication and authorization procedure is completed and the sync request is accepted - may send an acceptance message to the requesting handheld (605 in Figure 6) and then start the proxy sync handler (606 in Figure 6) to process the synchronization.

[0075] Figure 8 depicts an example of a detailed illustration of a proxy sync handler. As depicted, the proxy sync handler may first receive from the requesting handheld device the name of the application to be synchronized and the three IDs respectively for the replica host, the sync logic, and the transformation code (801). The information can be received directly from the handheld device that makes the sync request, or through a directory server as described earlier. If there is no more applications to sync, the handheld may send a termination message to indicate so. If such a termination message is received instead of the aforementioned three IDs, the sync handler terminates the synchronization process. Otherwise, the sync handler may request the cache manager to retrieve the sync logic based on the sync logic ID (802). An example of the cache management is discussed in Figure 9. After the sync logic is retrieved, the sync handler may then establish a connection to the replica of this application based on the replica host ID (803). Once the connection is established, the sync handler can start executing the sync logic to perform the synchronization task for this application (804). A more detailed example of the execution of the sync logic will be discussed with reference to Figure 10.

[0076] Figure 9 depicts an example, of the cache manager deployed by the synchronization proxy to manage the caching and retrieval of objects such as the sync logic and transformation codes for handheld applications. Upon receiving a request for an object (901), the cache manager checks to see if this object is in the cache. The cache can be either in main memory (303) or in disks (302).

[0077] If the requested object is not in the cache, the cache manager retrieves this object from the network based on the ID of this object (902). Once the object is retrieved from the network, it is inserted into the cache (903) and returned to the requester (904). If the object is found in the cache, the cache manager checks to see if the object in the cache is still current. If it is not, the cache manager retrieves it from the network using its ID (902). After the up-to-date object is retrieved from the network, it replaces the older one in the cache (903) and is returned to the requester (904). If the object is up to date, it is returned to the requester (904).

[0078] While cache object currency validation is not a feature of the present invention, those skilled in the art will appreciate that many such existing techniques can be adopted by the cache manger feature of the present invention. For example, a cache manager may adopt a policy such that no object is considered up to date if it is in the cache for over a fixed period of time. Another example is that a cache manager may register with the original object provider so that every time when an original object is updated (or created or deleted), the provider may send a message to the cache manager about the change of this object. The cache manager, upon receiving such a message, may then either remove this object from the cache, mark it as invalid, or retrieve the up-to-date one and replace the old one in the cache.

[0079] An example of the execution of the sync logic (804) by the proxy sync handler feature (Figure 8) of the present invention is illustrated in more detail in Figure 10. As depicted, the sync logic involves reading data from and writing data to both the handheld device (1003) making the sync request, and the replica host (1004) hosting the replica of the application of this sync

logic for the requesting handheld. There may be many differences between the data management systems in one handheld device and the information management system deployed in the replica host of this handheld device (e.g., PALM PILOTTM e-mail vs. Lotus NOTE-STM e-mail). For example, the data formats may be different. The storage capacities between the handheld device and the replica host may be different (e.g., PALM PILOTTM vs. workstation). The display devices between them may also be different (e.g., black-and-white low-resolution PALM PILOTTM screen vs. a high-resolution PC monitor). The need to properly transform information from one end (e.g., a workstation replica host) to the other (e.g., a PALM PILOTTM) is addressed in the present invention by the execution of the transformation code.

[0080] During the execution of the sync logic for an application (Figure 10), the sync logic determines the update direction for each data item of this application in the handheld and in the replica host. For example, for two data items of the same unique ID from the handheld and the replica host respectively, the sync logic may determine that the handheld data item overrides the replica host one, or vice versa. The sync logic may also determine that these two data items are the same and no data movement is necessary. It is also possible that the sync logic may find that these two items conflict (e.g., both have been updated independently) and decide to duplicate each item while retaining its own version.

[0081] In the proxy sync handler feature of the present invention, before a data item is to be written from one device to another (e.g., from the replica host to the handheld device, or vice versa), the proxy sync handler checks to see if a data transformation is applicable. If it is, the proxy sync handler can request the transformation code (specified by the ID sent from the handheld device making the sync request) from the cache manager (1002). The cache manager may process the procedures illustrated in Figure 9 to make available the requested object. Once the transformation code for the application being synchronized is available, the proxy sync handler executes the code for the data item to be written to the device (1003).

[0082] Figure 11 depicts an example of logic for a data manager (207) or a replica manager (406). The proxy sync handler (306 and Figure 8) can use conventional techniques to read data from (1103) and write data to (1104) both the handheld devices and their replica hosts through APIs. It is up to the data manager (207) and the replica manager (406) to interpret the API calls received from the sync proxy and to perform the appropriate data or replica management functions. Those skilled in the art will appreciate that these API calls can be typical data management functions such as read a data item (1103), write a data item (1104), update a data item (1105), delete a data item, etc. In order to process synchronization more efficiently, the sync logic for an application can be written to take advantage of any version and update history information available from the handhelds and their replica hosts in order to expedite the synchronization process. Those skilled in the art will appreciate that the aforementioned APIs can include functions that retrieve (1106) and/or set (1106) version and update history information from the handhelds and the replica hosts.

[0083] Figure 12 depicts an example of the replica acceptor feature of the present invention (405).

[0084] The replica acceptor can be a looping process in which it first waits for an API call (1201). When an API call arrives from the sync proxy, the replica acceptor may start an authentication and authorization process to verify the identity of the requesting sync proxy (1202). If the authentication and authorization process fails, the replica acceptor rejects the API call (1203).

[0085] Otherwise, the replica acceptor starts the replica manager (1202) to process the appropriate replica management function.

[0086] Figure 13 depicts an example of the load manager feature in a sync proxy of the present invention (308). When the proxy sync acceptor (Figure 6) receives a sync request and detects that the this sync proxy is overloaded, it can start the load manager to forward this sync request to another sync proxy (603). The load manager, upon receiving a demand to forward a sync request (1301), determines if there is another proxy on the same network that is not overloaded (1302). If one is found, the load manager forwards the sync request to this sync proxy (1304). If every proxy on the same network has a full load, the load manager rejects this sync request (1303). Those skilled in the art may appreciate that the load managers from all the sync proxies may participate in a protocol to exchange load information from one another. Based on this load information from other proxies on the same network, a load manager may also adopt a policy in choosing the proxy to forward (e.g., the load manager may choose the proxy that has the current lowest computation load) for overall load balancing over the network.

[0087] The proxy sync handler may decide to forward a sync request based not just on its load condition as illustrated in (602). It is possible that in a network deploying multiple sync proxies, each sync proxy specializes in data synchronization for a specific handheld device or information management system. For example, in a network supporting data synchronization for handheld devices that include a PALM PILOTTM and a WINDOWS CETM based device, two sync proxies are deployed with proxy 1 specializing in data synchronization for PALM PILOTTM and proxy 2 for data synchronization for Windows CE. Thus, based on this configuration, it is likely that more synchronization logic and transformation codes associated with PALM PILOT devices will be cached by proxy 1 and more synchronization logic and transformation codes associated with Windows CETM devices, will be cached by proxy 2 as

time goes by. Upon receiving a sync request (601), a sync proxy instead of just checking the load (602), may also check for the device type and, if it does not specialize in this device type, it may forward the request to the proxy specializing in this device type. For example, proxy 1, upon receiving a sync request from a Windows CETM device, may forward this request to proxy 2 which specializes in data synchronization for Windows CETM and therefore is more likely to have cached up-to-date sync logic and transformation codes needed to perform this sync request.

[0088] Instead of a pre-configured specialization division scheme as described above, the sync proxies may among themselves broadcast their caching situation dynamically, similar to the load condition that they share among each other. This way, each sync proxy can dynamically forward a sync request to another sync proxy where the caching condition is more favorable for processing this request.

[0089] A preferred embodiment of the present invention includes features that can be implemented as software tangibly embodied on a computer program product or program storage device for execution on a CPU (201, 301) provided with the client (101, 102) and server (105-107). For example, software implemented in a popular object-oriented computer executable code such as Sun's JAVATM provides portability across different platforms. Those skilled in the art will appreciate that other procedure-oriented and object-oriented (OO) programming environments, including but not limited to C++ and Smalltalk can also be employed.

[0090] Those skilled in the art will also appreciate that methods of the present invention may be implemented as software for execution on a computer or other processor-based device. The software may be embodied on a magnetic, electrical, optical, or other persistent program and/or data storage device, including but not limited to: magnetic disks, DASD, bubble memory; tape; optical disks such as CD-ROMs and DVD (digital video disks); and other persistent (also called nonvolatile) storage devices such as core, ROM, PROM, flash memory, or battery backed RAM. Those skilled in the art will appreciate that one or more of the components instantiated in the memory (203) of the client (101,102) or server (105-107) could be accessed and maintained directly via disk (302), storage memory (202), the network (108), another server, or could be distributed across a plurality of servers.

[0091] A sync proxy or sync server logic, process and apparatus employing sync logic and/or transformation codes distributed in the network to increase the scalability and improve the manageability of synchronization between a client such as: an Internet appliance; a handheld or palmtop device; a smart phone or intelligent pager; and their remote replica sources. An example of an adaptive sync server for synchronizing data between a client and a remote host replica, which stores a replica of data on the client includes sync acceptor logic for identifying a replica host and a sync logic which is application specific to a data type associated with the client and the remote host; wherein the sync logic can be located anywhere on a network remote to the sync server and the remote host, in response to a sync request; and sync handler logic, coupled to the sync acceptor logic, for retrieving remote sync logic from the network, connecting to the remote host, and synchronizing the data between the client and the remote host using the retrieved sync logic. The sync request could explicitly (or by default) identify the replica host and the sync logic or could include an address of a directory service storing information for identifying the replica host and the sync logic. The sync request could also or alternatively include information for identifying a (local or remote) device specific transformation code for converting data between the client and the remote host during synchronization; and means for retrieving the transformation code based on the information and converting the data between the client and the remote host based on the device specific transformation code.

Claims

1. An adaptive sync server for synchronizing data between a client and a remote host replica, which stores a replica of data on the client, comprising:

   sync acceptor logic for identifying a replica host and a sync logic which is application specific to a data type associated with the client and the remote host; wherein the sync logic can be located anywhere on a network remote to the sync server and the remote host, in response to a sync request; and

   sync handler logic, coupled to the sync acceptor logic, for retrieving remote sync logic from the network, connecting to the remote and synchronizing the data between the client and the remote host using retrieved sync logic.

2. An adaptive server for transforming data between a client and a remote host replica, which stores a replica of data on the client, comprising:

   acceptor logic for identifying the remote replica host and a device specific transformation code for transforming the data on the remote host to that of a device type associated with the client, wherein the transformation code can be located anywhere on a network remote to the server and the remote host, in response to a request; and

   handler logic, coupled to the acceptor logic, for retrieving remote transformation code and transforming the data between the client and

the remote host based on the device specific transformation code.

3. A client device adapted for connection to a sync server wherein synchronization of data is performed between the client and a remote host replica storing a replica of the data on the client, comprising:

a central processing unit (CPU);

a memory, coupled to the CPU, storing executable code for execution on the CPU, said code comprising:

a sync initiator for communicating to a sync server a sync request for identifying the replica host and a sync logic which is application specific to a data type associated with the client and the remote host; wherein the sync logic can be located anywhere on a network remote to the sync server and the remote host; and

sync handler logic, coupled to the sync initiator logic, for communicating synchronized data with the remote host via the sync server; and

a data manager, coupled to the sync handler, for processing one or more of data management and replica management functions.

4. The client of claim 3, wherein the client is a disconnectable handheld device selected from a group consisting of handheld computer; a smart phone, or an intelligent pager, comprising means for establishing a connection to the network.

5. The client of claim 3, wherein the data management comprise one or more of reading a data item; writing a data item; updating a data item; and deleting a data item.

6. A client device adapted for connection to an adaptive server wherein transformation of data is performed between the client and a remote host replica storing a replica of the data on the client, comprising:

a central processing unit (CPU);

a memory, coupled to the CPU, storing executable code for execution on the CPU, said code comprising:

an initiator for communicating to the server a request for identifying the replica host and a device specific transformation code for transforming data between the client and the remote host; wherein the transformation code can be located anywhere on a network remote to the server and the remote host; and

handler logic, coupled to the initiator logic, for communicating transformed data with the remote host via a remote server; and

a data manager, coupled to the handler logic, for processing one or more of data management and replica management functions.

7. In an adaptive sync server, a method for adaptively synchronizing data between a client and a remote host replica storing a replica of data on the client, the method comprising the steps of:

identifying a replica host and a sync logic which is application specific to a data type associated with the client and the remote host; wherein the sync logic can be located anywhere on a network remote to the sync server and the remote host, in response to a sync request;

retrieving remote sync logic from the network based on the request; and

connecting to the remote host and synchronizing the data between the client and the remote host using retrieved remote sync logic.

8. In an adaptive server, a method for transforming data between a client and a remote host replica storing a replica of data on the client, further comprising the steps of :

identifying the remote host replica and a device specific transformation code for transforming the data on the remote host to that of a device type associated with the client, wherein the transformation code can be located anywhere on a network remote to the server and the remote host, in response to a request; and

retrieving remote transformation code, in response to the request; and

transforming the data between the client and the remote host based on the device specific transformation code.

9. A program storage device readable by a machine, tangibly embodying a program of instructions executable by a sync server to perform method steps for adaptively synchronizing data between a client and a remote host replica storing a replica of data on the client, said method steps comprising:

identifying a replica host and a sync logic which is application specific to a data type associated with the client and the remote host; wherein the sync logic can be located anywhere on a network remote to the sync server and the remote host, in response to a sync request;

retrieving remote sync logic from the network; and

connecting to the remote host and synchronizing the data between the client and the remote host using retrieved remote sync logic.

10. A program storage device readable by a machine, tangibly embodying a program of instructions executable by a server to perform method steps for adaptively transforming data between a client and a remote host replica storing a replica of data on the client, said method steps comprising:

identifying the remote host replica and a device specific transformation code for transforming the data on the remote host to that of a device type associated with the client, wherein the transformation code can be located anywhere on a network remote to the server and the remote host, in response to a request; and

retrieving remote transformation code; and

transforming the data between the client and the remote host based on the device specific transformation code.

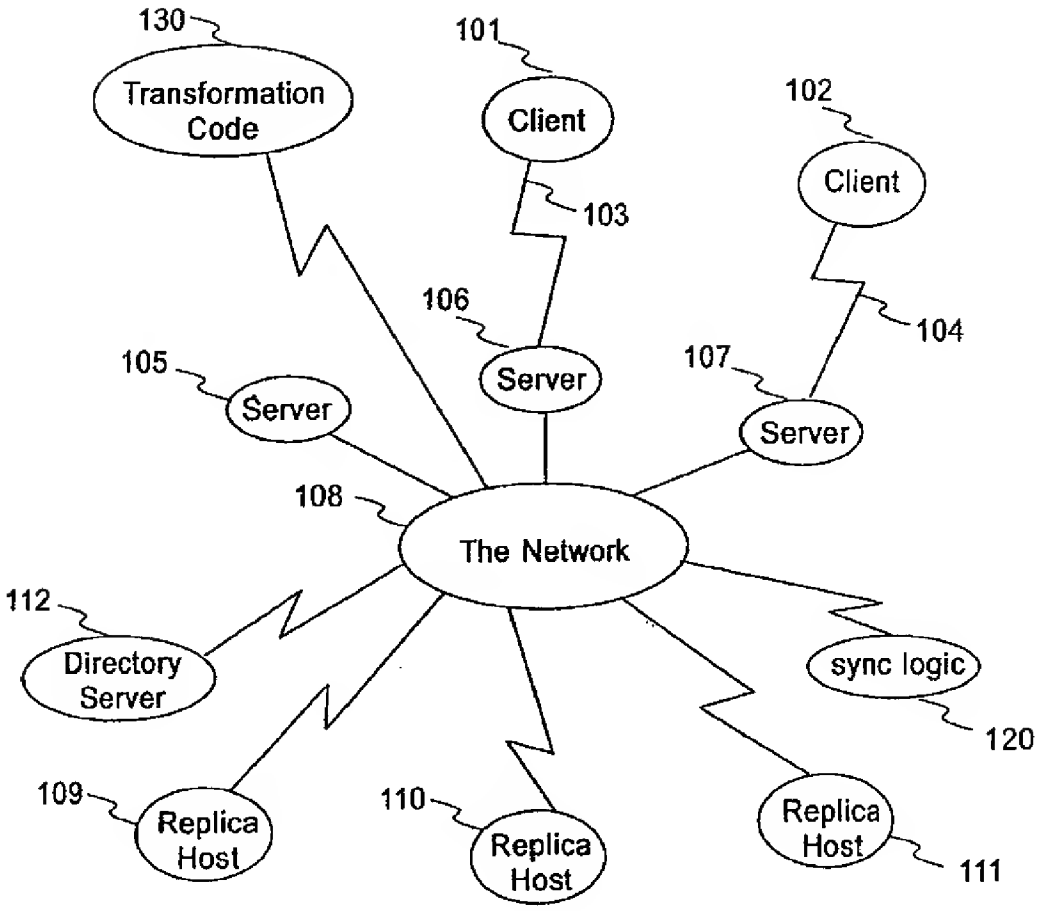
130
Transformation Code
101
Client
103
102
Client
104
106
Server
105
Server
107
Server
108
The Network
112
Directory Server
sync logic
120
109
Replica Host
110
Replica Host
Replica Host
111

Fig.1

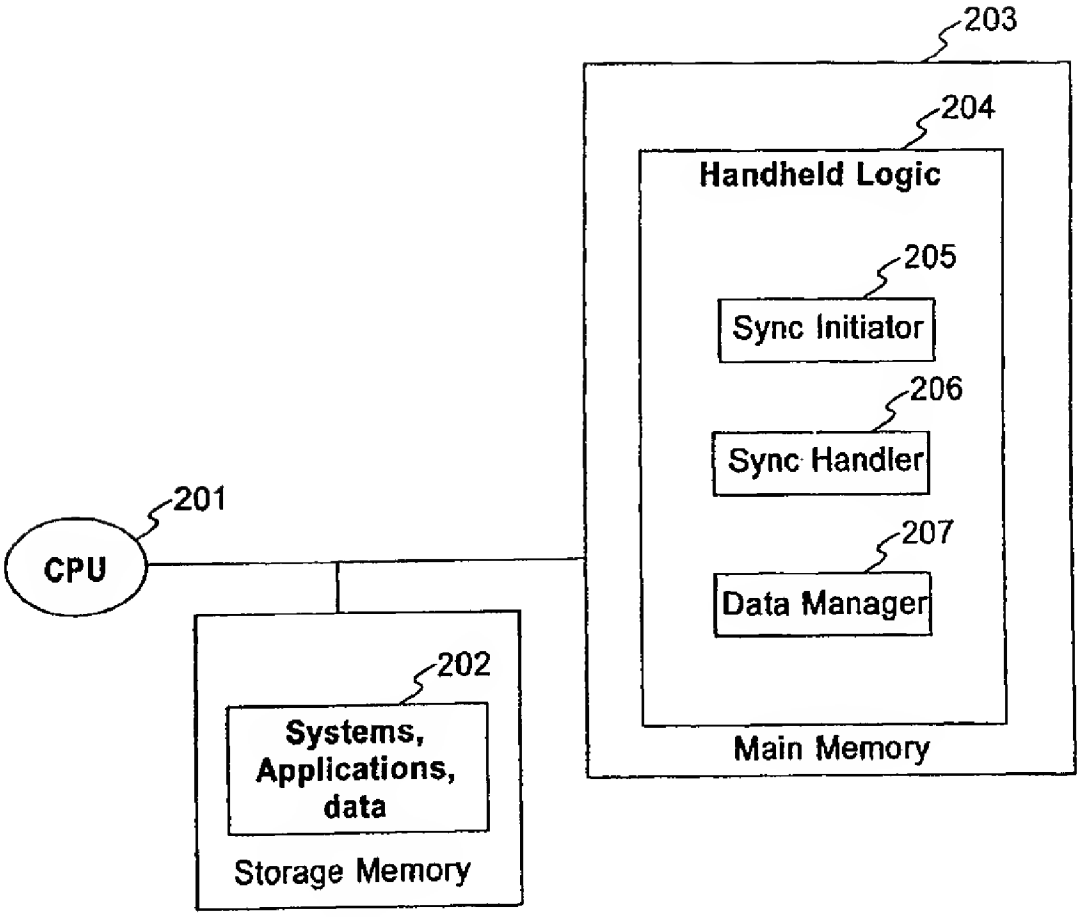
203
204
Handheld Logic
205
Sync Initiator
206
Sync Handler
207
Data Manager
Main Memory
201
CPU
202
Systems, Applications, data
Storage Memory

Fig.2

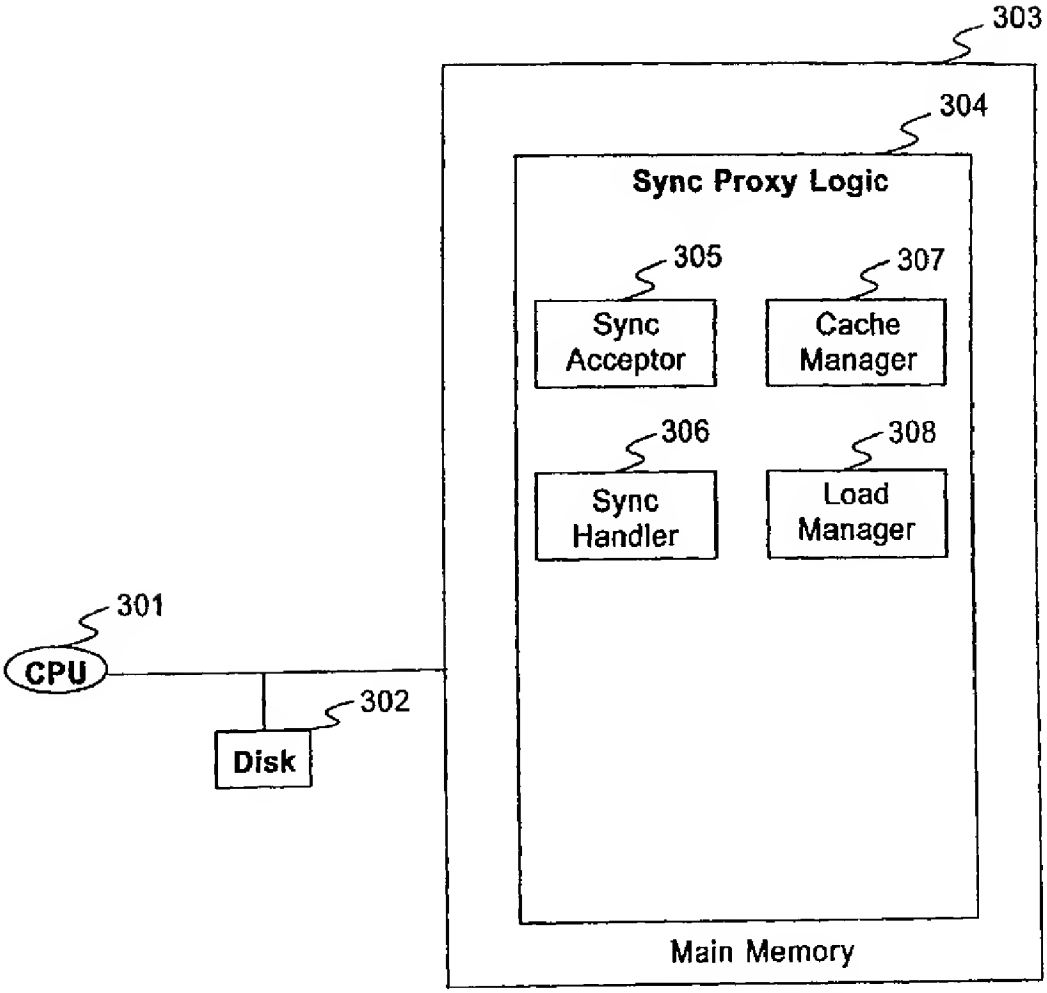
303
304
Sync Proxy Logic
305
Sync Acceptor
307
Cache Manager
306
Sync Handler
308
Load Manager
301
CPU
302
Disk
Main Memory

Fig.3

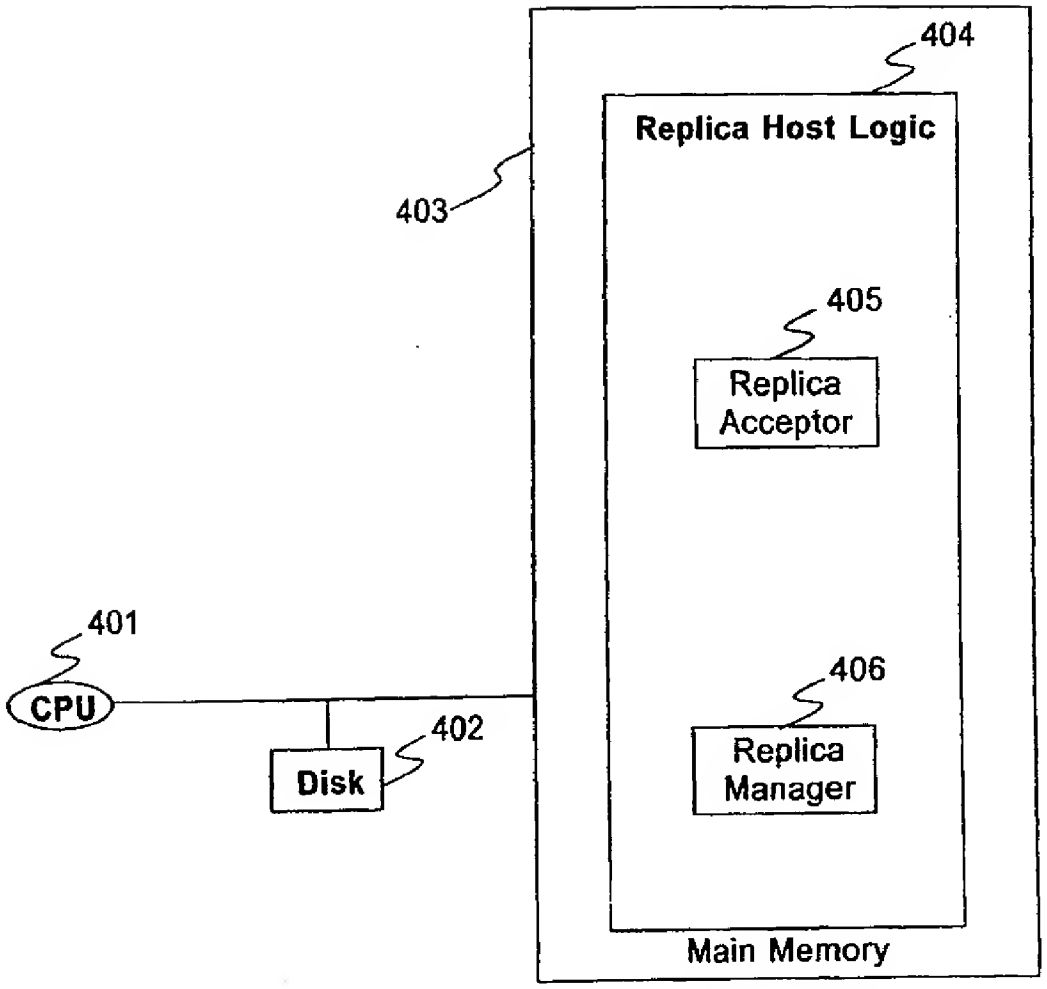
404
Replica Host Logic
403
405
Replica Acceptor
401
CPU
406
Replica Manager
402
Disk
Main Memory

Fig.4

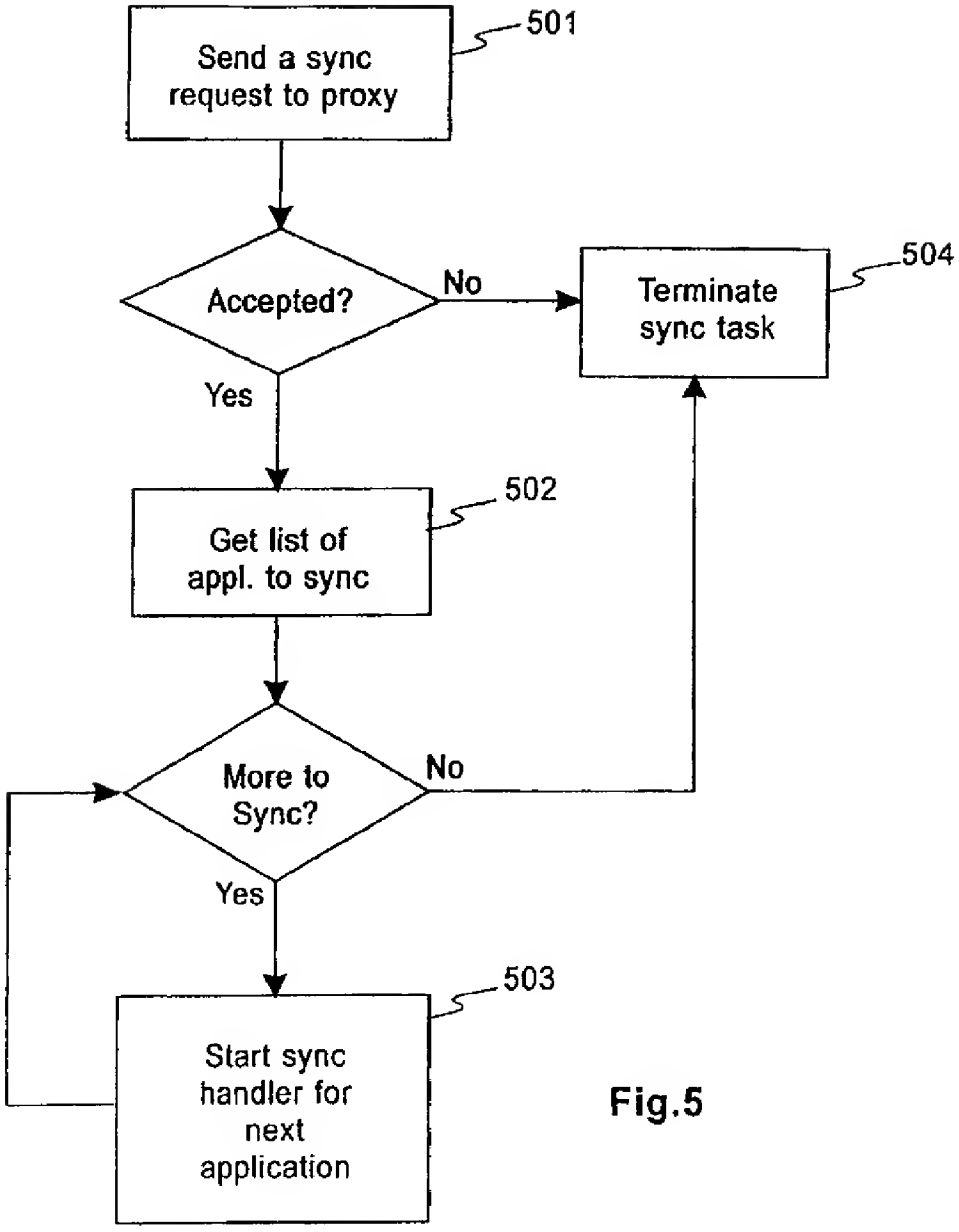
Send a sync request to proxy
501
Accepted?
No
Terminate sync task
504
Yes
Get list of appl. to sync
502
More to Sync?
No
Yes
Start sync handler for next application
503

Fig.5

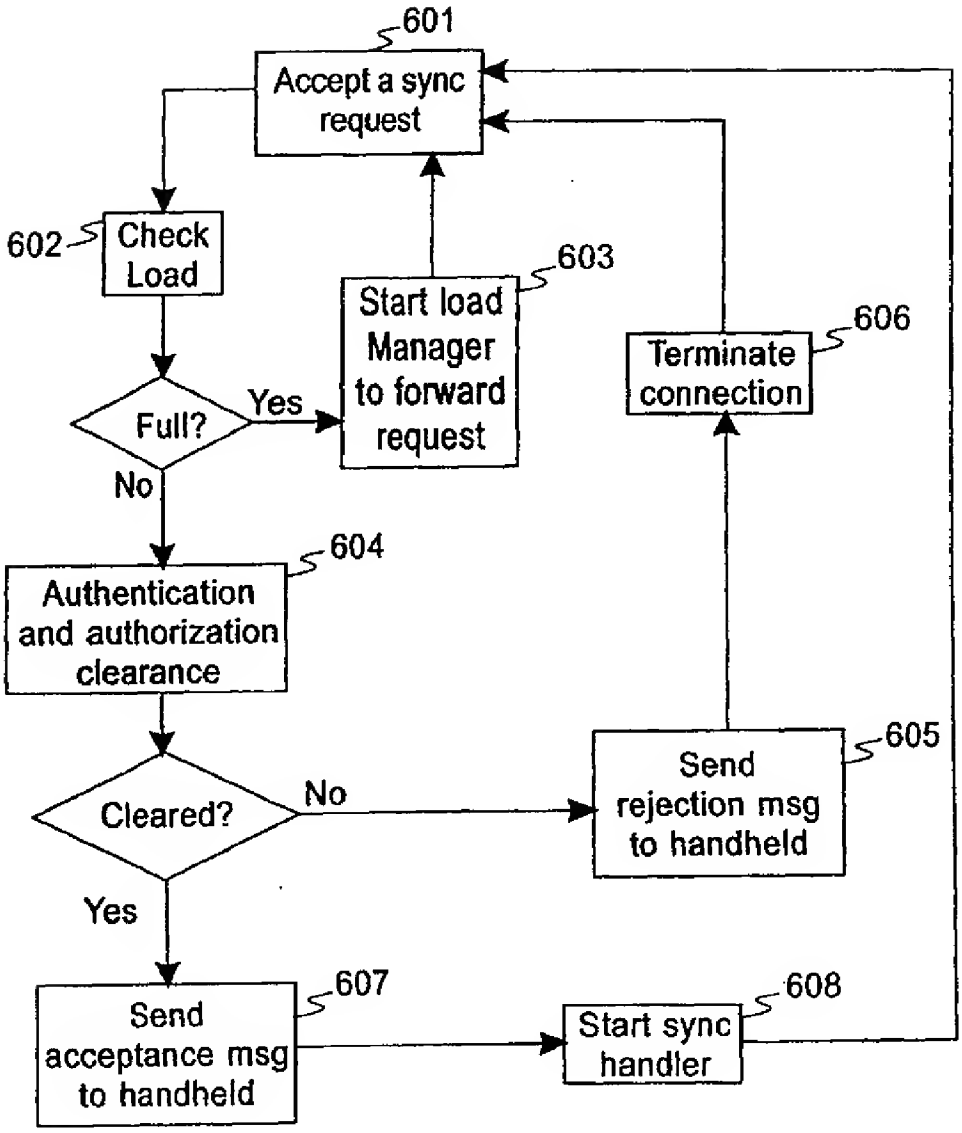
601
Accept a sync request
602
Check Load
603
Start load Manager to forward request
606
Terminate connection
Full?
Yes
No
604
Authentication and authorization clearance
Cleared?
No
605
Send rejection msg to handheld
Yes
607
Send acceptance msg to handheld
608
Start sync handler

Fig.6

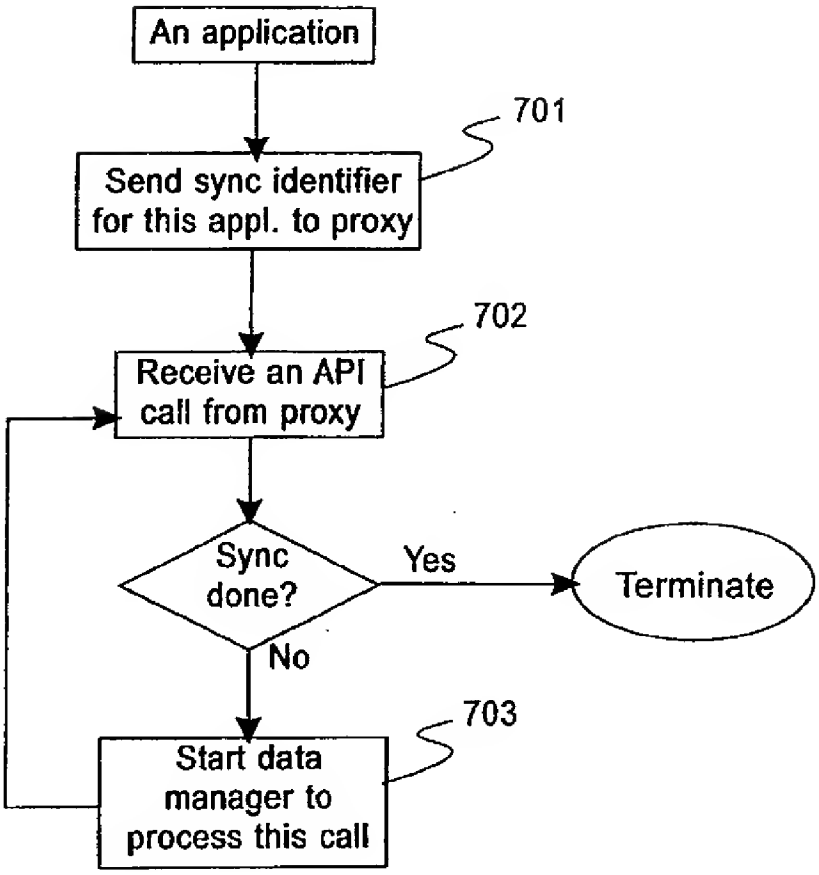
An application
701
Send sync identifier for this appl. to proxy
702
Receive an API call from proxy
Sync done?
Yes
Terminate
No
703
Start data manager to process this call

Fig. 7

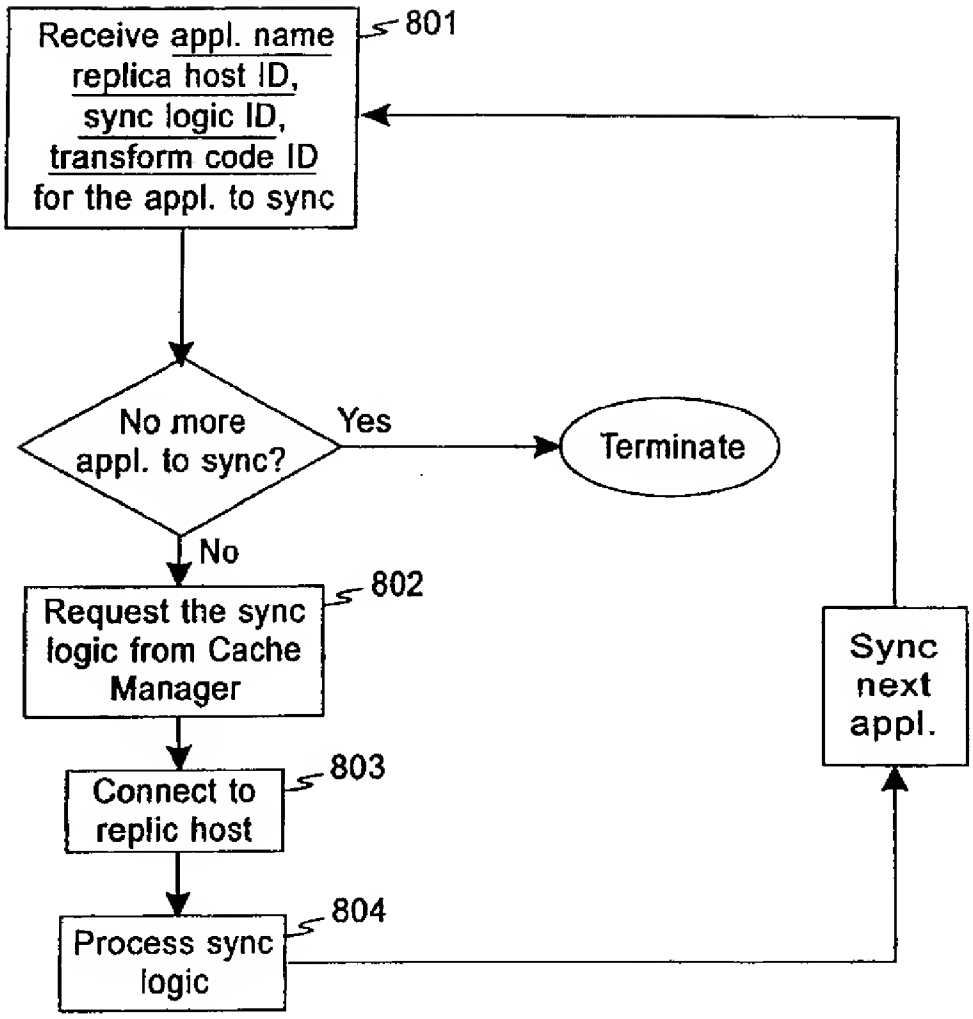
Receive appl. name
replica host ID,
sync logic ID,
transform code ID
for the appl. to sync
801
No more
appl. to sync?
Yes
Terminate
No
Request the sync
logic from Cache
Manager
802
Connect to
replic host
803
Process sync
logic
804
Sync
next
appl.

Fig.8

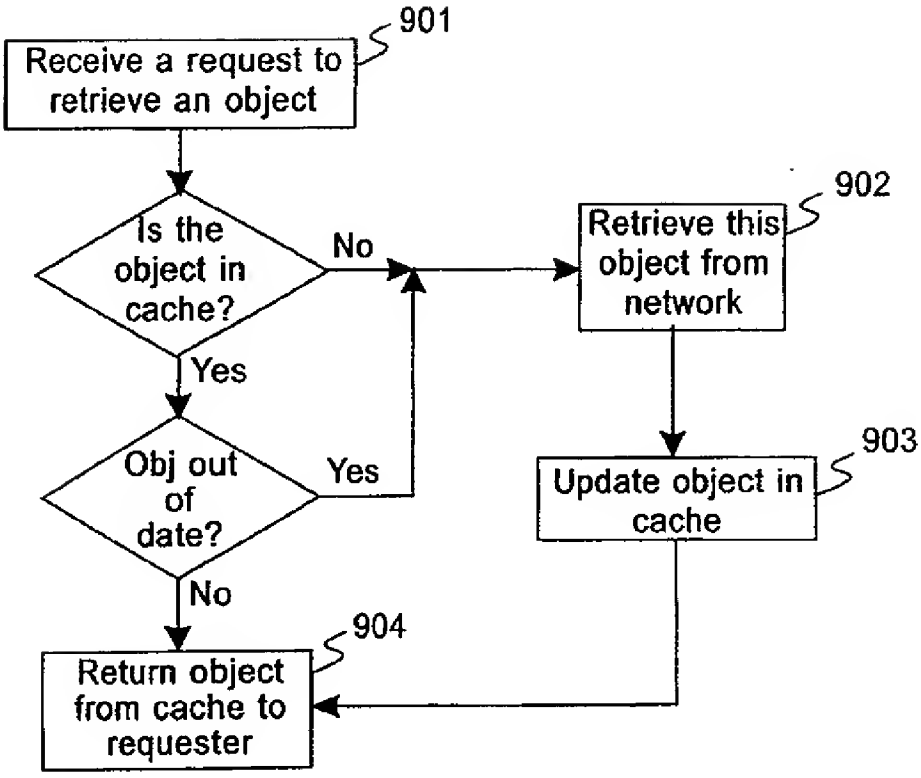
901
Receive a request to retrieve an object
Is the object in cache?
No
902
Retrieve this object from network
Yes
Obj out of date?
Yes
903
Update object in cache
No
904
Return object from cache to requester

Fig.9

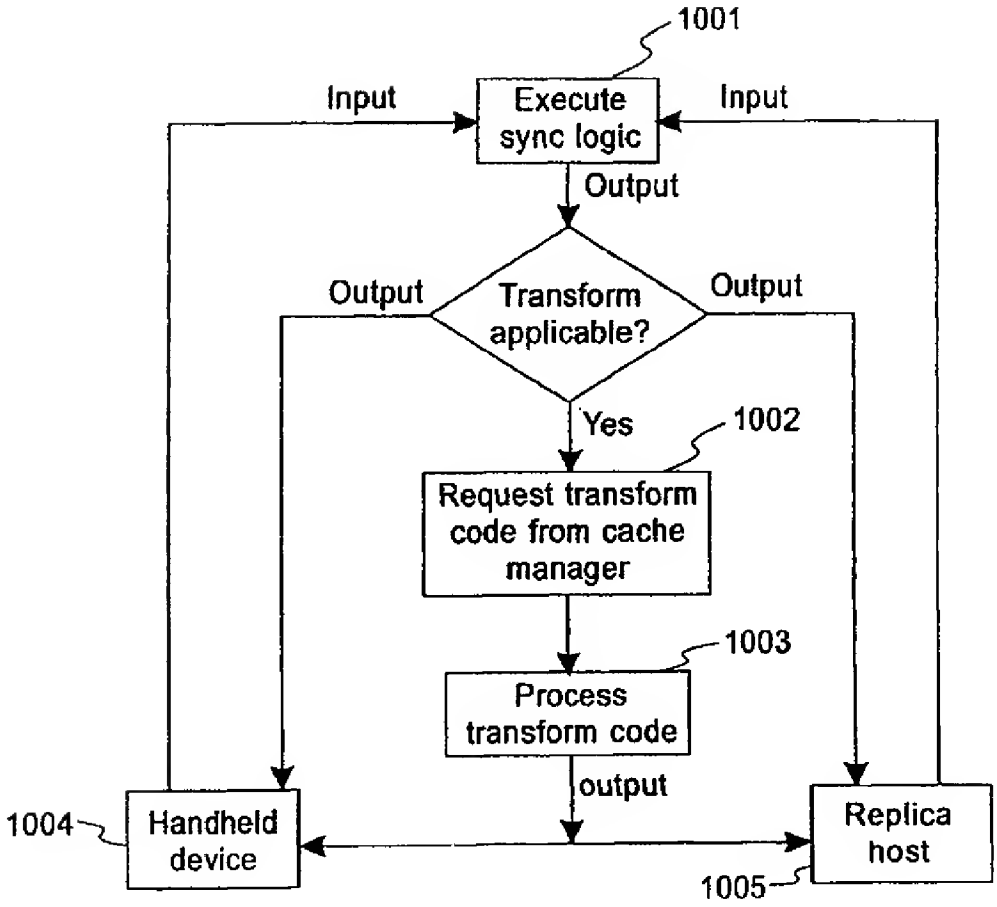
1001
Input
Execute sync logic
Input
Output
Output
Transform applicable?
Output
Yes
1002
Request transform code from cache manager
1003
Process transform code
output
1004
Handheld device
Replica host
1005

Fig.10

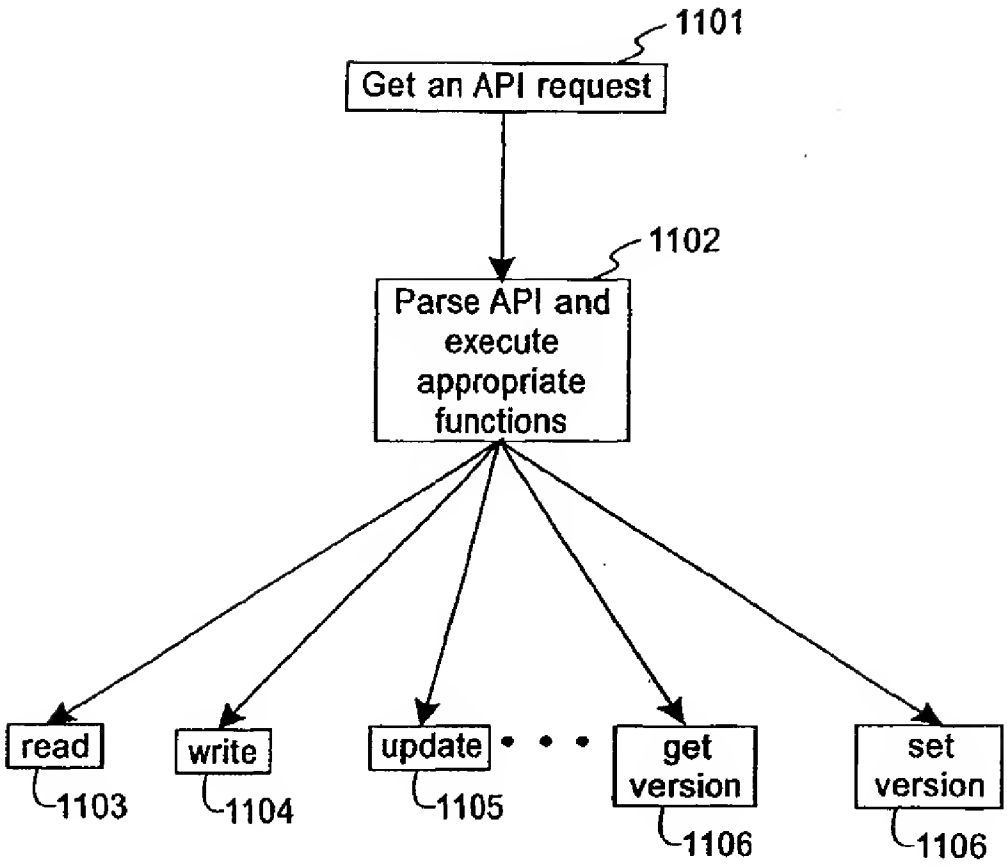
1101
Get an API request
1102
Parse API and execute appropriate functions
read
1103
write
1104
update
1105
get version
1106
set version
1106

Fig.11

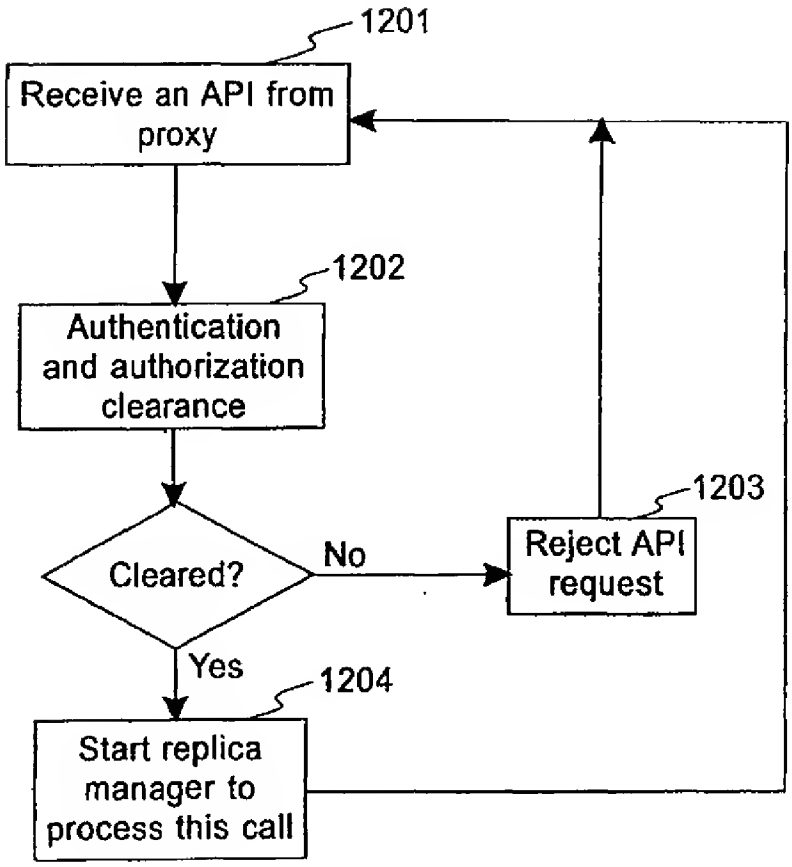
1201
Receive an API from proxy
1202
Authentication and authorization clearance
Cleared?
No
1203
Reject API request
Yes
1204
Start replica manager to process this call

Fig.12

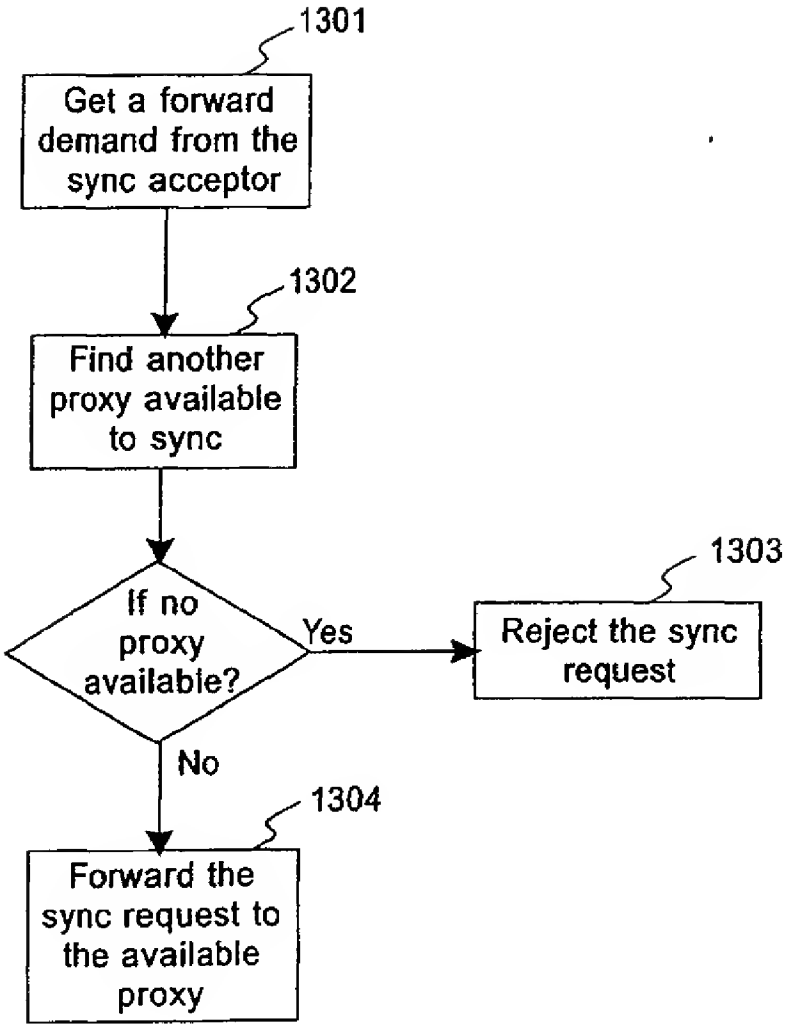
1301
Get a forward demand from the sync acceptor
1302
Find another proxy available to sync
If no proxy available?
Yes
1303
Reject the sync request
No
1304
Forward the sync request to the available proxy

Fig.13